Tutorial: Learning Deep Architectures

Yoshua Bengio, U. Montreal
Yann LeCun, NYU

ICML Workshop on Learning Feature Hierarchies,
June 18th, 2009, Montreal
Deep Motivations

- Brains have a deep architecture
- Humans organize their ideas hierarchically, through composition of simpler ideas
- Unsufficiently deep architectures can be exponentially inefficient
- Distributed (possibly sparse) representations are necessary to achieve non-local generalization
- Intermediate representations allow sharing statistical strength
Deep Architecture in the Brain

- Retina
- Area V1
- Area V2
- Area V4

- Pixels
- Edge detectors
- Primitive shape detectors
- Higher level visual abstractions
Deep Architecture in our Mind

- Humans organize their ideas and concepts hierarchically
- Humans first learn simpler concepts and then compose them to represent more abstract ones
- Engineers break-up solutions into multiple levels of abstraction and processing
Architecture Depth

Element set:
- *
- sin
- +
- -

Inputs:
- x
- a
- b

Depth = 4

Output

Element set:
- neuron
- neuron
- neuron
- ...
- neuron

Inputs:
- neuron
- neuron
- neuron

Depth = 3
Good News, Bad News

Theoretical arguments: deep architectures can be

2 layers of

\[
\begin{align*}
\text{logic gates} & \quad \text{formal neurons} \\
\text{RBF units} & \quad = \text{universal approximator}
\end{align*}
\]

Theorems for all 3:
(Hastad et al 86 & 91, Bengio et al 2007)

Functions representable compactly with k layers may require exponential size with k-1 layers
The Deep Breakthrough

- Before 2006, training deep architectures was unsuccessful, except for convolutional neural nets


Greedy Layer-Wise Pre-Training

Stacking Restricted Boltzmann Machines (RBM) → Deep Belief Network (DBN)
Stacking Auto-Encoders
Greedy Layerwise Supervised Training

Generally worse than unsupervised pre-training but better than ordinary training of a deep neural network (Bengio et al. 2007).
Supervised Fine-Tuning is Important

- Greedy layer-wise unsupervised pre-training phase with RBMs or auto-encoders on MNIST
- Supervised phase with or without unsupervised updates, with or without fine-tuning of hidden layers
Denoising Auto-Encoder

- Corrupt the input
- Reconstruct the uncorrupted input

KL(reconstruction | raw input)

Hidden code (representation)
Denoising Auto-Encoder

- Learns a vector field towards higher probability regions
- Minimizes variational lower bound on a generative model
- Similar to pseudo-likelihood
Stacked Denoising Auto-Encoders

- No partition function, can measure training criterion
- Encoder & decoder: any parametrization
- Performs as well or better than stacking RBMs for unsupervised pre-training
Deep Architectures and Sharing Statistical Strength, Multi-Task Learning

- Generalizing better to new tasks is crucial to approach AI
- Deep architectures learn good intermediate representations that can be shared across tasks
- A good representation is one that makes sense for many tasks
Why is Unsupervised Pre-Training Working So Well?

- Regularization hypothesis:
  - Unsupervised component forces model close to $P(x)$
  - Representations good for $P(x)$ are good for $P(y \mid x)$

- Optimization hypothesis:
  - Unsupervised initialization near better local minimum of $P(y \mid x)$
  - Can reach lower local minimum otherwise not achievable by random initialization
  - Easier to train each layer using a layer-local criterion
Learning Trajectories in Function Space

- Each point a model in function space
- Color = epoch
- Top: trajectories w/o pre-training
- Each trajectory converges in different local min.
- No overlap of regions with and w/o pre-training
Unsupervised learning as regularizer

- Adding extra regularization (reducing # hidden units) hurts more the pre-trained models

- Pre-trained models have less variance wrt training sample

- Regularizer = infinite penalty outside of region compatible with unsupervised pre-training
Better optimization of online error

- Both training and online error are smaller with unsupervised pre-training.
- As the number of samples approaches infinity, training error = online error = generalization error.
- Without unsupervised pre-training, we cannot exploit the capacity to capture complexity in the target function from the training data.
Learning Dynamics of Deep Nets

- As weights become larger, get trapped in basin of attraction ("quadrant" does not change)

- Initial updates have a crucial influence ("critical period"), explain more of the variance

- Unsupervised pre-training initializes in basin of attraction with good generalization properties
Restricted Boltzmann Machines

- The most popular building block for deep architectures
- Main advantage over auto-encoders: can sample from the model
- Bipartite undirected graphical model. x=observed, h=hidden

\[ P(x, h) = \frac{1}{Z} e^{-\text{Energy}(x, h)} = \frac{1}{Z} e^{b^T h + c^T x + h^T W x} \]

- \( P(h \mid x) \) and \( P(x \mid h) \) factorize: Convenient Gibbs sampling \( x \rightarrow h \rightarrow x \rightarrow h \ldots \)
- In practice, Gibbs sampling does not always mix well
Boltzmann Machine Gradient

\[ P(x) = \frac{1}{Z} \sum_h e^{-\text{Energy}(x,h)} = \frac{1}{Z} e^{-\text{FreeEnergy}(x)} \]

- Gradient has two components: ‘positive phase’ and ‘negative phase’

\[ \frac{\partial \log P(x)}{\partial \theta} = - \frac{\partial \text{FreeEnergy}(x)}{\partial \theta} + \sum \tilde{x} P(\tilde{x}) \frac{\partial \text{FreeEnergy}(x)}{\partial \theta} \]

\[ = - \sum_h P(h|x) \frac{\partial \text{Energy}(x)}{\partial \theta} + \sum \tilde{x}, \tilde{h} P(\tilde{x}, \tilde{h}) \frac{\partial \text{Energy}(x)}{\partial \theta} \]

- In RBMs, easy to sample or sum over \( h \mid x \):
- Difficult part: sampling from \( P(x) \), typically with a Markov chain
Training RBMs

- **Contrastive Divergence (CD-k):** start negative Gibbs chain at observed $x$, run $k$ Gibbs steps.

- **Persistent CD (PCD):** run negative Gibbs chain in background while weights slowly change

- **Fast PCD:** two sets of weights, one with a large learning rate only used for negative phase, quickly exploring modes

- **Herding** (see Max Welling’s ICML, UAI and workshop talks)
Deep Belief Networks

- **Sampling:**
  - Sample from top RBM
  - Sample from level k given k+1


- **Training:**
  - Variational bound justifies greedy layerwise training of RBMs
  - How to train all levels together?
Deep Boltzmann Machines

(Salakhutdinov et al, AISTATS 2009, Lee et al, ICML 2009)

- Positive phase: variational approximation (mean-field)

- Negative phase: persistent chain
  - Guarantees (Younes 89,2000; Yuille 2004)
  - If learning rate decreases in 1/t, chain mixes before parameters change too much, chain stays converged when parameters change.

- Can (must) initialize from stacked RBMs

- Salakhutdinov et al improved performance on MNIST from 1.2% to .95% error

- Can apply AIS with 2 hidden layers
Level-local learning is important

- Initializing each layer of an unsupervised deep Boltzmann machine helps a lot
- Initializing each layer of a supervised neural network as an RBM helps a lot
- Helps most the layers further away from the target
- Not just an effect of unsupervised prior
- Jointly training all the levels of a deep architecture is difficult
- Initializing using a level-local learning algorithm (RBM, auto-encoders, etc.) is a useful trick
Estimating Log-Likelihood

- RBMs: requires estimating partition function
  - Reconstruction error provides a cheap proxy
  - $\log Z$ tractable analytically for $< 25$ binary inputs or hidden
  - Lower-bounded with Annealed Importance Sampling (AIS)

- Deep Belief Networks:
  - Extensions of AIS (Salakhutdinov et al 2008)
Open Problems

- Why is it difficult to train deep architectures?
- What is important in the learning dynamics?
- How to improve joint training of all layers?
- How to sample better from RBMs and deep generative models?
- Monitoring unsupervised learning quality in deep nets?
- Other ways to guide training of intermediate representations?
- Getting rid of learning rates?
THANK YOU!

- Questions?
- Comments?